Douik *et al. EURASIP Journal on Advances in Signal Processing* (2016) 2016:1
DOI 10.1186/s13634-015-0293-z

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

Instantly decodable network coding for real-time device-to-device communications

Ahmed Douik[1*], Sameh Sorour[2], Tareq Y. Al-Naffouri[2,3] and Mohamed-Slim Alouini[3]

Abstract

This paper studies the delay reduction problem for instantly decodable network coding (IDNC)-based device-to-device (D2D) communication-enabled networks. Unlike conventional point-to-multipoint (PMP) systems in which the wireless base station has the sufficient computation abilities, D2D networks rely on battery-powered operations of the devices. Therefore, a particular emphasis on the computation complexity needs to be addressed in the design of delay reduction algorithms for D2D networks. While most of the existing literature on IDNC directly extend the delay reduction PMP schemes, known to be NP-hard, to the D2D setting, this paper proposes to investigate and minimize the complexity of such algorithms for battery-powered devices. With delay minimization problems in IDNC-based systems being equivalent to a maximum weight clique problems in the IDNC graph, the presented algorithms, in this paper, can be applied to different delay aspects. This paper introduces and focuses on the reduction of the maximum value of the decoding delay as it represents the most general solution. The complexity of the solution is reduced by first proposing efficient methods for the construction, the update, and the dimension reduction of the IDNC graph. The paper, further, shows that, under particular scenarios, the problem boils down to a maximum clique problem. Due to the complexity of discovering such maximum clique, the paper presents a fast selection algorithm. Simulation results illustrate the performance of the proposed schemes and suggest that the proposed fast selection algorithm provides appreciable complexity gain as compared to the optimal selection one, with a negligible degradation in performance. In addition, they indicate that the running time of the proposed solution is close to the random selection algorithm.

Keywords: Device-to-device communications, Instantly decodable network coding, Maximum weight clique

1 Introduction

Network coding (NC), initiated by Ahlswede, Cai, Li, and Yeung in their seminal paper [1], is a simple yet powerful technique to improve the performance of the communication systems significantly. The fundamental concept of NC is to perform arbitrary coding operations on the contents of packets rather than the direct replication and forwarding implemented in traditional store-and-forward networks. Such inter-network coding enables a high throughput [2], a fast recovery [3], and reliable communications over lossy channels [4]. The merits of NC are even more noticeable in broadcast wireless networks allowing it to be a graceful solution and a suitable technique for real-time applications [5, 6].

In the last few years, instantly decodable network coding (IDNC) attracted a considerable amount of research [7–10]. As its name indicates, IDNC allows instantaneous and progressive decoding of packets, a propriety of great interest for real-time applications. IDNC is implemented by mixing packets using the binary field $\mathbb{F}_2$, i.e., encoding is performed using binary XOR operations [11]. To further reduce the decoding complexity, non-instantly decodable packets are discarded at users [12]. Such features in IDNC enable the design of cost-efficient receiver devices as they allow fast XOR-based encoding and decoding and remove the need for buffers.

Consider a radio access network in which a base station (BS) is required to deliver a frame, composed of several packets, to a set of users. In an initial phase, the BS broadcasts the packets one after the other. The erasure nature of the wireless medium creates a diversity of received/lost packets at each user. The “sender" takes advantage of such diversity to transmit XOR-encoded

*Correspondence: ahmed.douik@caltech.edu
[1] California Institute of Technology (Caltech), Pasadena, California, USA
Full list of author information is available at the end of the article

packets targeting multiple receivers. The process, called the recovery phase, is repeated until all users successfully receive all packets. In previously mentioned IDNC works, the BS is assumed to be the sole sender in the system and hence the only responsible for both the initial and the recovery phases. Such point-to-multipoint (PMP) network configuration jeopardizes the ability of the BS to satisfy the ever more increasing data rate requirements notably in the next generation mobile radio system (5G) [13]. To mitigate the aforementioned effect, the notion of device-to-device (D2D) communications [14] is proposed as a promising technique for 5G. Taking advantages of the potentially more reliable short-range communications [14], devices are allowed to transmit to each other coded packet resulting in an increase in performance [15–18].

Despite its obvious advantages, the emergence of the D2D paradigm entails the redesign of the old protocols instead of their mere transplantation as it brings an additional set of challenges, particularly in terms of the computation complexity. Whereas in conventional PMP systems, the BS has the sufficient computation abilities to carry sophisticated optimizations, and D2D networks rely on battery-powered operations of the devices. Therefore, a particular emphasis on the computation complexity needs to be addressed in the design of delay reduction algorithms for D2D networks. Such redesign ranges from the optimization steps to the sacrifice of optimality to the benefit of convergence speed. This paper proposes to reduce the complexity of the delay reduction algorithms through efficient construction, update, and dimension reduction of the solution. The paper, further, suggests a fast selection algorithm with appreciable complexity gain as compared to the literature schemes and a negligible degradation in performance.

1.1 Related work

Due to the erasure nature of the links in wireless networks that affects the delivery of meaningful data, the receivers are no longer able to decode the frame synchronously. Therefore, a better use of the channel and network does not reflect an adequate throughput at higher communication layers [19]. Studies on IDNC can mainly be divided into two groups in which the delay is considered as the following:

- Completion time: the overall transmission time
- Decoding delay: the individual delay when delivered a useless packet at its reception moment

The completion time experienced is composed of a fixed delay (the initial transmission phase of the frame) and a variable delay (the recovery period). This definition of delay depends on the channel condition. If the channel condition is harsh, i.e., high erasure probability, the completion time increases regardless of the reception status of the packet. On the other hand, the decoding delay offers a definition more independent of the channel conditions since no delay is taken into account if the intended packet is erased and only delays due to the chosen encoded packet are considered. Furthermore, the decoding delay quantifies the degradation as compared to the "optimal" scheme in which the sender is able to satisfy all users by a new packet at each transmission.

In all aforementioned works, the authors considered the decoding delay as the sum of all the individual decoding delays experienced by all receivers. This definition of delay does not permit to have an equitable distribution of the delays between the different receivers since only the sum of all the individual delays counts. This paper introduces the maximum delay, i.e., the maximum value of the decoding delay experienced by all users, as a more reliable delay metric in IDNC that offers a better quality of service.

Recently, some works [20–23] begin to address the delay reduction problem in IDNC-enable D2D network configuration. While the authors in [20, 21, 23] design algorithm for the sum decoding delay reduction, references [20, 22] propose schemes for the completion time reduction. However, these works intersect as they directly extend the delay reduction PMP schemes, known to be NP-hard, to the D2D setting without considering the complexity issue for such battery-powered networks.

This paper's main contribution is to propose efficient algorithms for the delay reduction in IDNC-based D2D networks. This paper introduces and focuses on the maximum-delay reduction as it represents the most general solution. The paper reduces the complexity of the solution by first proposing efficient methods for the construction, the update, and the dimension reduction of the IDNC graph. The paper, further, shows that, under particular scenarios, the problem boils down to a maximum clique problem. Due to the complexity of discovering such maximum clique, the paper presents a fast selection algorithm.

The techniques proposed in this paper are independent of the setting and the considered metric. Indeed, as delay minimization problems in IDNC-based systems are equivalent to a maximum weight clique problems in the IDNC graph, the presented algorithms can be applied to different delay aspects and network settings. For example, they can be applied to reduce the complexity of the sum decoding delay minimization problem in imperfect feedback scenario proposed in [24] to the decentralized completion time reduction algorithm for D2D networks proposed in [22].

2 System model and problem formulation

2.1 System model and parameters

Consider a set $\mathcal{M}$ of M geographically close devices that require a set $\mathcal{N}$ of N source packets that a BS holds. In

this paper, the term packet has a general connotation that includes messages, files, frames from a video stream, etc. Each device $i \in \mathcal{M}$ is interested in receiving all the packets of the frame $\mathcal{N}$, regardless of the order.

In the first N time slots, the BS broadcasts the N source packets of the frame $\mathcal{N}$ uncoded. Each device listens to the transmitted packets and sends an acknowledgement (ACK) upon each successful reception. The ACK is communicated with the appropriate frequency and modulation so its reception is perfect. The channel between the BS and the ith device is modeled as a memory-less erasure channel with a packet erasure probability q_i assumed to be, at each transmission, constant and independent of the other channels. At the end of this phase, each packet of the frame is considered to be acknowledged by at least one device. Otherwise, such packet is re-transmitted by the BS until the assumption is satisfied. Therefore, for each device i, packets of the frame $\mathcal{N}$ can be in one of the following sets:

- The Has set $\mathcal{H}_i$: packets received by device i
- The Wants set $\mathcal{W}_i = \mathcal{N} \backslash \mathcal{H}_i$: packets erased at device i

This paper assumes single-hop transmissions, i.e., all devices are in the transmission range of each other. Therefore, the system does not require any additional feedback load as ACKs can be overheard by all devices. Similar to the BS-user ACKs, the reception of the user-to-user feedback is assumed to be perfect. Each device stores the information, concerning the diversity of received/lost packets, in a *state matrix* (SM) $\mathbf{S} = [s_{ij}]$, $\forall\, i \in \mathcal{M}$, $\forall\, j \in \mathcal{N}$ such that:

$$s_{ij} = \begin{cases} 0 & \text{if } j \in \mathcal{H}_i \\ 1 & \text{if } j \in \mathcal{W}_i. \end{cases} \tag{1}$$

After the initial transmissions, devices cooperate to recover their missing packets by transmitting to each other binary XOR-encoded packets of the ones they already hold. In the considered single-hop transmissions system, at each time slot, only one device is transmitting. The packet combination is chosen according to the diversity of lost/received packets represented in the SM and the expected erasure patterns of the links. Following the same channel model as the BS-device, let p_{ij}, $i,j \in \mathcal{M}$ denote the packet erasure probability of the memory-less channel from device j to device i. All the packet erasure probabilities are assumed to be known and constant during a single transmission. A packet combination, in the recovery phase, can be one of the following two options for each device i:

- *Instantly decodable:* a combination is instantly decodable for device i if it contains *exactly one* packet from $\mathcal{W}_i$.
- *Non-instantly decodable:* a combination is non-instantly decodable for device i if it does not contain exactly a single packet from $\mathcal{W}_i$.

The decoding delay [12] is defined as follows:

Definition 1. At any recovery phase transmission, a device i, with non-empty Wants set, increases by one unit if it successfully receives a packet that is non-instantly decodable. The maximum delay is the highest decoding delay experienced by all devices.

In other words, the decoding delay measures the number of non-instantly decodable packets received by each user which translates the degradation from the perfect scheme in which the sender is able to satisfy users with a new packet at each transmission.

2.2 Problem formulation

The decoding delay reduction problem is the one of selecting the communicating devices and the packet combinations that reduce the maximum delay of all devices. Due to the dynamic nature of the links, finding a schedule of transmission that optimally reduce the decoding delay for the entire recovery phase, prior to its start, is intractable. A commonly adopted technique [9, 10, 12, 20, 21, 23] is to minimize the decoding delay on-line, i.e., at each recovery transmission. $D_j(\kappa_i, n)$ is define as the total decoding delay experienced by device j from the beginning of the recovery phase until the transmission at time n in which device i is transmitting the packet combination κ_i. The on-line decoding delay reduction problem, at the nth transmission, can be written as follows:

$$\min_{i \in \mathcal{M}} \left\{ \min_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left[\max_{j \in \mathcal{M}} \left(\mathbb{E}(D_j(\kappa_i, n)) \right) \right] \right\}, \tag{2}$$

where the notation $\mathcal{P}(\mathcal{X})$ refers to the power set of the set $\mathcal{X}$ and $\mathbb{E}$ refers to the expectation operator.

The following lemma reformulates the maximum-delay reduction problem in a more tractable form:

Lemma 1. *The on-line decoding delay reduction problem (2) can be reformulated by the following more tractable expression:*

$$\min_{i \in \mathcal{M}} \left\{ \min_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left[\mathbb{P}(\mathbb{X}(\kappa_i, n)) \right] \right\}, \tag{3}$$

where $\mathbb{P}$ *is the probability operator and* $\mathbb{X}(\kappa_i, n)$ *is the event that the maximum delay increases at the nth transmission, i.e.,*

$$\mathbb{X}(\kappa_i, n) = \left(\max_{j \in \mathcal{M}} D_j(n-1) < \max_{j \in \mathcal{M}} D_j(\kappa_i, n) \right).$$

Proof. Let i^* and κ_i^* be the transmitting user and packet combination that optimally solve (2). Furthermore, let j^* and $j^\dagger$ be defined as the users that maximize the decoding delay after sending the combinations κ_i and κ_i^*. In other words,

$$\max_{j\in\mathcal{M}} \mathbb{E}(D_j(\kappa_i, n)) = \mathbb{E}(D_{j^*}(\kappa_i, n)) \tag{4}$$

$$\max_{j\in\mathcal{M}} \mathbb{E}(D_j(\kappa_i^*, n)) = \mathbb{E}(D_{j^\dagger}(\kappa_i^*, n)) \tag{5}$$

To show the lemma, it is sufficient to show that the combination of the transmitting user i^* and the packet κ_i^* achieves the minimal value of the optimization problem (3).

By inspection of the on-line decoding delay $D_j(\kappa_i, n)$ in (2), the transmitted packet combination κ_i affects only the most recent increase in the decoding delay. Hence, such on-line decoding delay can be decomposed as $D_j(\kappa_i, n) = d_j(\kappa_i, n) + D_j(n-1)$, where $d_j(\kappa_i, n)$ is the delay experienced by device j for the transmission of the packet combination κ_i and $D_j(n-1)$ is the cumulative decoding delay experience until the nth transmission excluded. Therefore, $D_j(\kappa_i, n)$ is Bernoulli random variable that takes the values $D_j(n-1)$ and $D_j(n-1)+1$ with probabilities α and $1-\alpha$ where α is a parameter that depends on the transmitting user and packet combination. Therefore, the expected decoding delay can be written as:

$$\begin{aligned}\mathbb{E}(D_j(\kappa_i, n)) &= \alpha D_j(n-1) \\ &\quad + (1-\alpha)(D_j(n-1)+1) \\ &= D_j(n-1) + 1 - \alpha. \end{aligned} \tag{6}$$

Given that (i^*, κ_i^*) is the optimal solution to (2), the following inequality holds for any combination (i, κ):

$$\begin{aligned}\max_{j\in\mathcal{M}} \mathbb{E}(D_j(\kappa_i, n)) &\geq \max_{j\in\mathcal{M}} \mathbb{E}(D_j(\kappa_i^*, n)) \\ \mathbb{E}(D_{j^*}(\kappa_i, n)) &\geq \mathbb{E}(D_{j^\dagger}(\kappa_i^*, n)) \\ D_{j^*}(n-1) + 1 - \alpha^* &\geq D_{j^\dagger}(n-1) + 1 - \alpha^\dagger \end{aligned} \tag{7}$$

where α^* and $\alpha^\dagger$ are the probabilities of the Bernoulli random variables of users j^* and $j^\dagger$ to increase the maximum delay. Since both users j^* and $j^\dagger$ can potentially increase the decoding delay at the nth transmission, they have the same value of the cumulative decoding delay $D_{j^*}(n-1) = D_{j^\dagger}(n-1)$. Finally,

$$1 - \alpha^* \geq 1 - \alpha^\dagger \tag{8}$$

To conclude the proof, it is sufficient to note that the probability that the maximum-delay increase can be written as follows:

$$\begin{aligned}&\mathbb{P}\left(\max_{j\in\mathcal{M}} D_j(n-1) < \max_{j\in\mathcal{M}} D_j(\kappa_i, n)\right) \\ &\mathbb{P}\left(D_j^*(n-1) < D_j^*(\kappa_i, n)\right) = 1 - \alpha^*. \end{aligned} \tag{9}$$

Similarly, $\mathbb{P}\left(\mathbb{X}\left(\kappa_i^*, n\right)\right) = 1 - \alpha^\dagger$. Therefore, using the inequality illustrated in (8), it can easily be seen that $\left(i^*, \kappa_i^*\right)$ is an achievable lower bound for the optimization problem (3). Finally, we conclude that both optimization problems (2) and (3) are equivalent. □

The rest of the paper presents efficient methods to optimally and heuristically solve the optimization problem (3). In the next section, the probability distributions of the maximum-delay increase are derived. Such expressions allow the construction of the IDNC graph and the reformulation of the optimal packet selection problem as a maximum weight clique search that can be globally solved using efficient algorithms, e.g., [25, 26]. To further reduce the complexity of the proposed solution, the paper suggests optimizing the construction, the update, and the dimension of the graph. Finally, the paper presents a fast device selection algorithm so as to decrease both the complexity and the signaling in the system.

3 Maximum decoding delay reduction

This section illustrates the optimal solution to the optimization problem (3). First, the maximum-delay increment are expressed for an arbitrary transmitting device and a feasible packet combination. Such expressions allow the reformulation of the packet combination problem as a maximum weight clique problem in the IDNC graph. Searching for the maximum weight clique for each device identifies the optimal transmitting device and the optimal packet combination to be transferred. Finally, with the optimal solution of (3) being potentially not unique, a multi-layer solution is suggested so as to improve the performance of the system by an efficient choice of one of the optima.

3.1 Maximum-delay increment

As the cumulative decoding delay is an increasing function, then for any transmitting device i and packet combination κ_i, it is clear that $D_j(n-1) \leq D_j(\kappa_i, n)$, $\forall\, j \in \mathcal{M}$, $\forall\, n > 1$. Therefore, the probability of a decoding delay increase can be reformulated as follows:

$$\begin{aligned}\mathbb{P}(\mathbb{X}(\kappa_i, n)) &= \mathbb{P}\left(D^*(n-1) < \max_{j\in\mathcal{M}} D_j(\kappa_i, n)\right) \\ &= 1 - \mathbb{P}\left(D^*(n-1) = \max_{j\in\mathcal{M}} D_j(\kappa_i, n)\right), \end{aligned} \tag{10}$$

where $D^*(n-1) = \max_{j\in\mathcal{M}} D_j(n-1)$ is the maximum delay reached in the $(n-1)$th transmission.

Define $\mathcal{L}(n)$ as the set of devices having the maximal decoding delay at time $n-1$, i.e., $i \in \mathcal{L}(n) \Leftrightarrow D_i(n-1) = D^*(n-1)$). As the decoding delay increase by, at most, one unit at each transmission, then devices $j \notin \mathcal{L}(n)$ have zero probability to increase the maximum delay as compared to $D^*(n-1)$ even if they experience a decoding delay. Hence,

the probability (10) is controlled solely by devices $j \in \mathcal{L}(n)$ and may be written as follows:

$$\begin{aligned}\mathbb{P}(\mathbb{X}(\kappa_i, n)) &= 1 - \mathbb{P}\left(d_j(\kappa_i, n) = 0, \forall\, j \in \mathcal{L}(n)\right) \\ &= 1 - \prod_{j \in \mathcal{L}(n)} \mathbb{P}(d_j(\kappa_i, n) = 0). \end{aligned} \tag{11}$$

According to Definition 1, the delay $d_j(\kappa_i, n)$ experienced by device j, with non-empty Wants set, at the nth transmission by device i of the packet combination κ_i can be expressed as follows:

$$\mathbb{P}\left(d_j(\kappa_i, n)) = 0\right) = \begin{cases} 1 & \text{if } j \text{ targeted by } \kappa_i \\ p_{ij} & \text{otherwise,} \end{cases} \tag{12}$$

where a device j is said to be targeted by the combination κ if it is instantly decodable for him.

Let $\tau(\kappa)$ be the set of targeted devices by the packet combination κ and let M_w be the set of devices having non-empty Wants set. Substituting (12) in (11), the probability of event $\mathbb{X}(\kappa_i, n)$ to occur can be expressed as follows:

$$\mathbb{P}(\mathbb{X}(\kappa_i, n)) = 1 - \prod_{j \in (\mathcal{L}(n) \cap M_w) \setminus \tau(\kappa_i)} p_{ij}. \tag{13}$$

3.2 Local IDNC graph

To determine both all possible XOR-based combinations and the devices that can instantly code each of them in a PMP system, the authors in [12] introduce the IDNC graph. This subsection extends the formulation to the D2D setting under investigation. While the BS in a PMP network can generate any packet combination, devices, in a D2D configuration, can produce combination using only the packets they have already received in previous transmissions. This subsection illustrates how an arbitrary transmitting device i can build a similar IDNC graph. In the context of D2D communications, such graph is referred to as the *local* IDNC graph.

The local IDNC graph $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i)$ of device i is constructed by creating a vertex $v_{jk} \in \mathcal{V}_i$ for each device $j \neq i$ and each packet $k \in \mathcal{W}_j \cap \mathcal{H}_i$. The intersection of the packets with the Has set of device i, i.e., $k \in \mathcal{H}_i$, ensures that the device holds all the packets of the combination. Two distinct vertices v_{jk} and v_{lm} are connected with an edge $e_{jk,lm} \in \mathcal{E}_i$ if one of the two following conditions is true:

- C1: $k = m \Rightarrow$ packet k is needed by both devices j and l
- C2: $k \in \mathcal{H}_l$ and $m \in \mathcal{H}_j \Rightarrow$ the packet combination $k \oplus m$ is instantly decodable for both devices j and l

Given the local IDNC graph $\mathcal{G}_i$ as constructed above, the following lemma links the set of possible packet combinations to the set of cliques of the graph.

Lemma 2. *The set of all feasible packet combinations that device i can generate is represented by the set of maximal cliques in $\mathcal{G}_i$. Applying binary XOR to the vertices of a selected maximal clique κ_i produces the packet combination that targets the devices $\tau(\kappa_i)$ represented by that maximal clique.*

Proof. The proof of this lemma is omitted as it mirrors the steps used in proving Proposition 1 in [27]. In fact, as device i can generate combination using only the packets it already holds, then the whole network can be considered as if it contains $\mathcal{M}' = \mathcal{M} \setminus \{i\}$ devices and $\mathcal{N}' = \mathcal{H}_i$ packets. Following a similar steps as in [27], the reduced network yields the desired result. □

3.3 Maximum decoding delay reduction

In order to reach the optimal solution to the optimization problem (3), the optimal packet combination that an arbitrary device i can generate in the nth transmission is first derived, i.e., the optimal solution to the following problem:

$$\min_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left[\mathbb{P}(\mathbb{X}(\kappa_i, n))\right]. \tag{14}$$

The optimal solution to the optimization problem (14) is characterized in the following theorem.

Theorem 1. *The optimal packet combination κ_i^* device i that can generate in the nth transmission to minimize the maximum delay is the maximum weight clique in its local IDNC graph $\mathcal{G}_i$ in which the weight of a vertex v_{jk} is the following:*

$$w_{jk}^* = \begin{cases} -\log(p_{ij}) & \text{if } j \in \mathcal{L}(n) \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Proof. Let κ_i^* be the optimal packet combination that device i can generate in the nth transmission. Substituting expression (13) in the optimization problem (14) yields the following expression of κ_i^*:

$$\begin{aligned}\kappa_i^* &= \arg \min_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left\{1 - \prod_{j \in (\mathcal{L}(n) \cap M_w) \setminus \tau(\kappa_i)} p_{ij}\right\} \\ &= \arg \max_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left\{\prod_{j \in (\mathcal{L}(n) \cap M_w) \setminus \tau(\kappa_i)} p_{ij}\right\} \\ &= \arg \min_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left\{\prod_{j \in \mathcal{L}(n) \cap \tau(\kappa_i)} p_{ij}\right\} \\ &= \arg \max_{\kappa_i \in \mathcal{P}(\mathcal{H}_i)} \left\{\sum_{j \in \mathcal{L}(n) \cap \tau(\kappa_i)} -\log(p_{ij})\right\}. \end{aligned} \tag{16}$$

According to Lemma 2, the set of feasible packet combinations that device i can generate is represented by the set of maximal cliques in $\mathcal{G}_i$. Therefore, the optimization problem (17) may be written as:

$$\kappa_i^* = \arg\max_{\kappa_i \in \mathcal{C}_i} \left\{ \sum_{j \in \mathcal{L}(n) \cap \tau(\kappa_i)} -\log(p_{ij}) \right\}, \tag{17}$$

where $\mathcal{C}_i$ is the set of maximal cliques in the local IDNC graph $\mathcal{G}_i$ of device i. In other words, the optimal packet combination κ_i^* that device i can generate in the nth transmission is the maximum weight clique in $\mathcal{G}_i$ in which the weights of vertices are defined in (15). □

In virtue of Theorem 1, the optimal packet combination that each device can generate can be determined along with its corresponding weight, i.e., the objective function of the optimization problem (14). The optimal solution of the maximum-delay reduction problem (3) can be obtained by simply choosing the transmitting device as the one that achieves the maximum weight. Such transmitting device selection can be reached via one of the following schemes:

- Each device broadcasts the weight of its maximum weight clique and the device with the maximum weight transmits for the upcoming time slot. The algorithm complexity per device is $O(M^2N)$, and the broadcast load is M collision-free transmissions.
- Each device determines the maximum weight clique of all the other devices and the device with the maximum weight transmits. The algorithm complexity per device is $O(M^3N)$. However, no broadcast load is required.

3.4 Multi-layer solution

As shown in the previous subsection, the optimal transmitting device i^* is the one that maximizes $\max_{i \in \mathcal{M}} y\left(\kappa_i^*\right)$, where $y(.)$ is the objective function of (3) and κ_i^* is the optimal packet combination obtained from Theorem 1. This subsection improves upon the proposed solution by noting that the optimal solution may not be unique and suggesting an improvement based on an efficient choice of one of the optima of (14).

According to Theorem 1, the optimal packet combination is the maximum weight clique in the local IDNC graph of device i^*. However, since vertices $v \notin \mathcal{L}(n)$ have a zero weight, then the maximum weight clique may not be unique. To improve the performance of the proposed algorithm, this paper proposes prioritizing devices that are more likely to be in $\mathcal{L}(n+\delta), \delta > 0$. It is easy to see that a device j is potentially in $\mathcal{L}(n+\delta), \delta > 0$ if δ is the smallest integer such that $D_j(n-1) + \delta = D^*(n-1)$ where $D^*(n-1) = \max_{j \in \mathcal{M}} D_j(n-1)$ is the maximum delay at the $(n-1)$th transmission. In fact, such device j is in $\mathcal{L}(n+\delta)$ if he experiences δ successive decoding delay increases.

Based on the prioritization above, this subsection suggest dividing the local IDNC graph into layer of decreasing importance. Let $\mathcal{L}_\delta, \delta = 0, 1, \cdots$ be the layers of the graph containing only the vertices v_{jk} such that $j \in \mathcal{L}(n+\delta)$. Further, let $\mathcal{L}_\delta(\kappa)$ be the layer that contains vertices in $\mathcal{L}_\delta$ that are adjacent to all the vertices in κ. In the layered graph proposed above, the weight of a vertex v_{jk} is $w(v_{jk}) = -\log(p_{ij})$. The following corollary indicates an efficient choice of one of the optima of (14).

Corollary 1. *The optimal packet combination $\kappa_{i^*}^*$ can be obtained by solving the maximum weight clique in the first layer $\mathcal{L}_0 = \mathcal{L}(n)$ of the local IDNC graph of device i^*. To serve the maximum number of devices according to their prioritization, the maximum weight clique problem is sequentially solved for layers $\mathcal{L}_\delta(\kappa), \delta = 1, \cdots$ containing the vertices connected to the maximum weight clique chosen so far. The steps are summarized in Algorithm 1.*

Algorithm 1 Multi-Layer Delay Reduction

```
Initialize κ*_{i*} = ∅
for δ = 0, 1, ··· do
    Construct L_δ(κ*_{i*}).
    Search κ the maximum weight clique in L_δ(κ*_{i*}).
    Set κ*_{i*} = κ*_{i*} ∪ κ
end for
```

Proof. It is clear that the algorithm described above respects the prioritization among devices as the maximum weight clique problem is solved sequentially for a decreasing order of priority. Therefore, it is sufficient to show that the set of vertices produced by Algorithm 1 is the maximum weight clique in the local IDNC graph.

Let κ be the set of vertices generated by Algorithm 1. It is easy to note that it is a clique since all the vertices are connected to each other. Further, assume that κ is not maximal, i.e., $\exists\, v \in \mathcal{V}_{i^*}$ such that $\kappa \cup v$ is a clique. Let $\mathcal{L}_\delta$ be the layer of vertex v. Since $\kappa \cup v$ is a clique, then v is connected to all vertices in κ and as a result $v \in \mathcal{L}_\delta(\kappa)$. Since the algorithm computes the maximum weight clique in $\mathcal{L}_\delta(\kappa)$, then either $v \in \kappa$ or v is not connected to κ. Both options are in contradiction with the original assumption. Therefore, such v does not exist. To conclude the proof, note that, in the local IDNC graph, only vertices in $\mathcal{L}(n)$ have a non-zero weight. The algorithm begins by searching the maximum weight clique in the $\mathcal{L}_0 = \mathcal{L}(n)$, then both approaches produce the same set of vertices in $\mathcal{L}(n)$. Therefore, κ is a maximum weight clique in the local IDNC graph and hence one of the optima of (14). □

4 Proposed algorithms

This section proposes reducing the complexity of finding the optimal solution to the maximum-delay reduction problem. The fundamental concept of such complexity reduction is improving the local IDNC graph formulation and the maximum weight clique search algorithm. Therefore, the first part of this section suggests an optimized method for the construction of such graph and its update. Afterwards, the paper derives sufficient conditions for reducing the size of the graph and transforming the maximum weight clique search to a simpler maximum clique discovery problem. Finally, the section proposes a heuristic to reduce further the complexity of the proposed solution and adapt the computation to the abilities of mobile devices.

4.1 IDNC graph construction and update

As shown in the previous section, the optimal solution to the maximum-delay reduction problem involves a maximum weight clique over the local IDNC graph of each device. Due to the update in the SM, such search requires the construction of the corresponding graph at each transmission. The complexity of generating the graph depends on the number of wanted packets by each device. However, it is always proportional to the quantity $\mathcal{O}(M^2N)$ where the proportionality constant is the number of desired packets, i.e., the number of 1's in the SM. This section proposes reducing such complexity by constructing an extended version of the local IDNC graph containing hidden vertices and updating it at each transmission.

The key idea of constructing the extended local IDNC graph of device i is to generate an IDNC graph similar to the BS's one. As a device is unable to target itself, its vertices are removed from the graph. Finally, vertices representing packets the device does not hold so far are hidden. Note that removing such vertices reduce the extended local IDNC graph to the one constructed in the previous section. Such hidden vertices are activated as soon as the device receives the corresponding packets. Therefore, the construction of the extended local IDNC graph $\mathcal{G}_i(\mathcal{V}_i, \mathcal{E}_i)$ of device i requires the generation of a vertex v_{jk} for each $j \in \mathcal{M}$ and $k \in \mathcal{W}_j$. The connectivity conditions between two vertices are the conditions C1 and C2 proposed in the previous section. Afterward, all vertices belonging to device i are removed, and all vertices representing packets that are not in the possession of device i (i.e., $v_{jk}|k \notin \mathcal{H}_i$) are set as hidden. The steps of the construction are summarized in Algorithm 2. Note that the complexity of generating the extended local IDNC graph is similar to the one of constructing the local IDNC graph, i.e., $\mathcal{O}(M^2N)$.

For a transmitted clique κ, only a subset κ^+ of the targeted devices in κ receives the combination due to possible erasures that may occur. In order to update the extended local IDNC graph, the considered devices overhear the acknowledgement to determine κ^+. Vertices in κ^+ being received are discarded from the graph, and the additional connections between the remaining vertices are added. Furthermore, if the device itself is targeted, then the hidden vertices corresponding to the packet he just received are activated, and the connections added. The complexity of updating the graph is proportional to $\mathcal{O}(MN)$ where the proportionality constant is the number of vertices in κ^+ which is clearly less than the number of 1's in the SM. The steps of the graph update are summarized in Algorithm 3.

Algorithm 2 Extended Local IDNC Graph Construction

```
Generate v_jk ∈ V_i for each j ∈ M and k ∈ W_j.
Generate E_i by connecting vertices using C1 and C2.
Set G_i = (V_i, E_i)
for all v_kl ∈ V_i do
  if k = i then
    Remove v_kl from G_i.
  end if
  if l ∉ H_i then
    Hide v_kl in G_i.
  end if
end for
```

Algorithm 3 Extended Local IDNC Graph Update

```
for all v_kl ∈ κ^+ do
  if k = i then
    Activate hidden v_jl, ∀ j ∈ M.
  else
    Remove v_kl from G_i.
    for all m ≤ M, m ≠ i do
      for all n ≤ N do
        if s_ml = 1 and s_kn = 1 and s_mn = 0 then
          Connect v_ml and v_kn.
        end if
      end for
    end for
  end if
end for
```

4.2 Complexity reduction

This section proposes reducing the complexity of finding the optimal solution to the maximum-delay reduction problem by deriving a sufficient condition to transform the maximum weight clique search to a maximum clique problem.

Assume the maximum clique in the local IDNC graph is unique, the following theorem states a sufficient condition under which the maximum weight clique problem is equivalent to a maximum clique problem that can be solved efficiently.

Theorem 2. *Let $w_{\max}$ and $w_{\min}$ be, respectively, the maximum and the minimum weight in the graph and let $\Delta_w = w_{\max} - w_{\min}$ be the weight range. A sufficient condition to transform the maximum weight clique search to a maximum clique problem is the following:*

$$|M_w| \leq w_{\max}/\Delta_w. \tag{18}$$

Proof. To prove this theorem, a sufficient condition is first derived in terms of the maximum clique size. To conclude the proof, an upper bound on the size of the maximum clique is established.

Let $\mathcal{C}$ be the maximum clique in the graph of size $|\mathcal{C}|$. Assume that the maximum weight clique $\mathcal{C}_w$ of size $|\mathcal{C}_w|$ is different from $\mathcal{C}$. Since the maximum clique is unique, then it is easy to infer that $|\mathcal{C}_w| < |\mathcal{C}|$. While the weight of $\mathcal{C}_w$ is upper bounded by $|\mathcal{C}_w| w_{\max}$, the weight of $\mathcal{C}$ is lower bounded by $|\mathcal{C}| w_{\min}$. Therefore, the difference in weight is upper bounded by the following quantity:

$$w(\mathcal{C}_w) - w(\mathcal{C}) \leq |\mathcal{C}_w| w_{\max} - |\mathcal{C}| w_{\min}. \tag{19}$$

For $\mathcal{C}_w$ to be the maximum weight clique, it should provide the highest weight among all cliques. In particular, its weight should be greater than that of $\mathcal{C}$. Therefore, a sufficient condition guaranteeing that such maximum weight clique does not exist is the following:

$$w(\mathcal{C}_w) - w(\mathcal{C}) \leq |\mathcal{C}_w| w_{\max} - |\mathcal{C}| w_{\min} \leq 0. \tag{20}$$

With the weights being positive quantities, the function $f(|\mathcal{C}_w|) = |\mathcal{C}_w| w_{\max} - |\mathcal{C}| w_{\min}$ is increasing in $|\mathcal{C}_w|$. Hence, verifying $\max_{|\mathcal{C}_w|<|\mathcal{C}|} f(|\mathcal{C}_w|) = f(|\mathcal{C}| - 1) \leq 0$ is sufficient to establish the results. Rearranging the terms of the former inequality yields the following condition:

$$|\mathcal{C}| \leq w_{\max}/\Delta_w. \tag{21}$$

The following lemma links the size of the maximum clique to the number of devices having non-empty Wants set:

Lemma 3. *The size of any clique in the local IDNC graph cannot exceed $|M_w|$. In particular, the size $|\mathcal{C}|$ of the maximum clique $\mathcal{C}$ verify $|\mathcal{C}| \leq |M_w|$.*

Proof. Assume $\exists\, \mathcal{C}$ clique in IDNC such that $|\mathcal{C}| > |M_w|$. Since all the vertices belongs to M_w, then in virtue of the pigeonhole principle $\exists\ v_{ij} \neq v_{ik} \in \mathcal{C}$. By construction of the graph, the packets belong to the Wants set, i.e., $j, k \in \mathcal{W}_i$. Therefore, the packet combination containing $j \oplus k$ is not instantly decodable for device i. However, the packet combination represented by the clique is instantly decodable for all the devices included in it. Thus, the size of any clique cannot exceed $|M_w|$. □

Substituting the result of Lemma 3 in the inequality (21) gives the sufficient condition (18) to transform the maximum weight clique search to a maximum clique problem: □

Furthermore, the following corollary derives a sufficient condition to transform the maximum weight clique problem to a maximum clique search for all the remaining transmissions:

Corollary 2. *Assuming that at the nth transmission, the following inequality is verified $|M_w| \leq \frac{w_{\min}}{\Delta_w}$, then for any transmission at time $m \geq n$, the maximum weight clique problem can be transformed to a maximum clique problem.*

Proof. According to Theorem 2, to prove the corollary, it is sufficient to show that for any time instant m, the following inequality holds $|M_w(m)| \leq \frac{w_{\max}(m)}{\Delta_w(m)}$ where the argument m is added to differentiate the different quantities at different transmissions. It is easy to infer that the property holds the nth transmission as shown in the following equality:

$$|M_w(n)| \leq \frac{w_{\min}(n)}{\Delta_w(n)} \leq \frac{w_{\max}(n)}{\Delta_w(n)}. \tag{22}$$

As the update of the graph of transmission to transmission consist of removing vertices, the following inequality can be easily inferred:

$$|M_w(m)| \leq |M_w(n)| \tag{23}$$

$$\Delta_w(m) \leq \Delta_w(n) \tag{24}$$

$$w_{\min}(n) \leq w_{\min}(m). \tag{25}$$

Combining the inequalities in (25) with (22) yields the following result:

$$|M_w(m)| \leq |M_w(n)| \leq \frac{w_{\min}(n)}{\Delta_w(n)} \leq \frac{w_{\min}(m)}{\Delta_w(m)}. \tag{26}$$

Using the same step as in (22), it is easy to conclude that $|M_w(m)| \leq \frac{w_{\max}(m)}{\Delta_w(m)},\ \forall\, m \geq n.$ □

4.3 Dimension reduction

As the complexity of finding the maximum weight clique heavily depends on the number of vertices in the graph, the subsection proposes to reduce the dimension of the

local IDNC graph. While the first part of the subsection reduces the number of devices, the rest reduces the number of packets.

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be the local IDNC graph and let i,j be the two devices such that $\mathcal{H}_i = \mathcal{H}_j$. The fundamental concept in reducing the number of devices is that such devices i,j can be served simultaneously. Therefore, instead of representing each by a vertex, this subsection suggest generating only a single vertex for both vertices. The weight of the vertex is defined as the sum of the individual weights of the vertices. The steps of the algorithm are illustrated in Algorithm 4. The following proposition characterizes the maximum weight clique in the reduced graph produced by Algorithm 4.

Algorithm 4 Local IDNC Graph Device Reduction

for all $i,j \in \mathcal{M}$ such that $\mathcal{H}_i = \mathcal{H}_j$ **do**
  for all $(v_{ik}, v_{jk}) \in \mathcal{V}^2$ **do**
    Remove v_{ik} and v_{jk} from $\mathcal{V}$.
    Generate $v_{\{i,j\}k}$ in $\mathcal{V}$.
    Set weight $w\left(v_{\{i,j\}k}\right) = w(v_{ik}) + w(v_{jk})$.
  end for
end for

Proposition 1. *The maximum weight clique in the reduced graph produced by Algorithm 4 is equivalent to the maximum weight clique in the local IDNC graph.*

Proof. To prove this proposition, it is sufficient to show that the vertices representing devices i and j have the same set of connections in the local IDNC graph and that they are connected to each other. In fact, having the same set of connection implies that if one of the vertices, e.g., v_{ik}, belongs to the maximum weight clique in the local IDNC graph, then the other, e.g., v_{jk}, also belongs and vice versa. The weight being the sum of the individual weights concludes the proof.

First, it is easy to infer that vertices v_{ik} and v_{jk} are connected since they satisfy the connectivity condition C1. Assume that a vertex v_{ik} is connected to a vertex v_{ml}. Two scenarios can occur:

- Vertices are connected through C1, i.e., $k = l$. Therefore, C1 is also verified for v_{jk} and v_{ml} and they are connected.
- Vertices are connected through C2, i.e., $l \in \mathcal{H}_i$ and $k \in \mathcal{H}_m$. Since $\mathcal{H}_i = \mathcal{H}_j$, then $l \in \mathcal{H}_j$. The condition $k \in \mathcal{H}_m$ still being valid; hence, C2 is also verified for v_{jk} and v_{ml} and they are connected.

Finally, vertices v_{ik} and v_{jk} have the same set of connections, and they are connected to each other which concludes the proof. □

Let $\mathcal{T}_k = \{i \in \mathcal{M} \mid k \in \mathcal{W}_i\}$ be the set of devices wanting packet k, and let k,l be the two packets such that $\mathcal{T}_k = \mathcal{T}_l$. The key idea in reducing the number of packets is that such packets k,l can never be served simultaneously. Therefore, instead of representing each by a vertex, this subsection suggest generating only one of them. The steps of the algorithm are illustrated in Algorithm 5. The following proposition characterizes the maximum weight clique in the reduced graph produced by Algorithm 5.

Algorithm 5 Local IDNC Graph Packet Reduction

for all $k,l \in \mathcal{N}$ such that $\mathcal{T}_k = \mathcal{T}_l$ **do**
  for all $(v_{ik}, v_{il}) \in \mathcal{V}^2$ **do**
    Remove v_{ik} (or v_{il}) from $\mathcal{V}$.
  end for
end for

Proposition 2. *The maximum weight clique in the reduced graph produced by Algorithm 5 is equivalent to the maximum weight clique in the local IDNC graph.*

Proof. To prove this proposition, it is sufficient to show that the vertices representing packets k and l have the same set of connections in the local IDNC graph and that they are not connected to each other. In fact, having the same set of connection implies that if one of the vertices, e.g., v_{ik}, belongs to the maximum weight clique in the local IDNC graph, then the other, e.g., v_{il}, cannot belong to it and vice versa. With both vertices having the same weight, the removal of packet k or packet l is arbitrary.

First, it is easy to infer that vertices v_{ik} and $v_{jl}, \forall\, i,j$ are not connected since they violate the connectivity conditions C1 and C2. Assume a vertex v_{ik} is connected to a vertex v_{mr} with $r \neq k$. Then, from the connectivity condition C2, the following hold $s_{ik} = s_{rm} = 1$ and $s_{mk} = s_{ri} = 0$. Since $\mathcal{T}_k = \mathcal{T}_l$, then it is clear that $s_{ik} = s_{il}$ and $s_{ml} = s_{mk}$. Substituting in the previous equalities yield $s_{il} = 1$ and $s_{ml} = 0$ which conclude that vertices v_{jl} and v_{mr} are connected. Since the weight depends solely on the device represented by the vertex rather than the packet, then the vertices v_{ik} and v_{il} have the same weight and removing any one of them is arbitrary. □

4.4 Fast selection algorithm

In order to perform fast device selection, this paper proposes a heuristic algorithm based on an approximation of the optimal solution. Instead of computing the optimal packet combination of each device and deciding afterwards which of the devices provides the highest gain, this subsection proposes calculating an upper bound on the achievable maximum-delay reduction. The transmitting

device is the one that possesses both most of the packets that allow to achieve the upper bound and the least cumulative decoding delay.

Assuming erasure-free transmissions, the optimal packet combination that reduces the maximum delay is the maximum clique $\tilde{\kappa}$ in the graph un-weighted $\mathcal{G}(\mathcal{V},\mathcal{E})$ containing only $\mathcal{L}(n)$. Each device can compute such packet combination by activating all the hidden vertices in its local IDNC graph and removing the weights and the vertices not belonging to $\mathcal{L}(n)$. Due to the difference between the erasure between devices and the limitation on the number of simultaneously targeted devices, the first layer $\mathcal{L}(n)$ is expected to contain only a few vertices; therefore, the determination of the maximum clique in this graph is cheap in terms of computation.

Choosing the transmitting device according to its ability to achieve the upper bound guarantee that the transmission is beneficial to the largest number of devices. However, as the transmitting device experiences an increase in the decoding delay, the decoding delay of such device should be considered in the selection process. Therefore, the transmitting device is the one that has the minimum decoding delay among the ones that have the maximum number of packets to achieve the upper bound. In other words, the device is selected according to the following optimization problem:

$$\begin{aligned} i^* &= \arg\max_{i\in\mathcal{M}} D_i(n) \\ \text{subject to} &|\mathcal{H}_{i^*}\cap\tilde{\kappa}| = \max_{i\in\mathcal{M}} |\mathcal{H}_i\cap\tilde{\kappa}|. \end{aligned} \tag{27}$$

If more than one device satisfy the upper bound with the minimum decoding delay, i.e., the optimal solution of (27) is not unique, then the one with the smallest index transmits. The steps of the fast selection algorithm are illustrated in Algorithm 6. The complexity of the proposed fast selection algorithm is $\mathcal{O}(MN)$ as compared to the $\mathcal{O}(M^2N)$ of the algorithm proposed in the previous section. It is also worth mentioning that the proposed fast selection algorithm does not have a broadcast load as all devices solve the optimization problem (27).

Algorithm 6 Fast Selection Algorithm

Generate un-weighted $\mathcal{G}(\mathcal{V},\mathcal{E})$ containing only $\mathcal{L}(n)$.
Solve $\tilde{\kappa}$ the maximum clique in $\mathcal{G}$.
Select device i^* using (27).
Transmit κ^* the maximum weight clique in $\mathcal{G}_{i^*}$.

5 Simulation results

This section presents the simulation results comparing the average maximum delay encountered by the devices while applying the point-to-multipoint system, the optimal, the fast, and the random device selections for D2D-enabled networks. The number of devices, packets, and erasure probabilities vary so as to study multiple scenario. In the random client selection algorithm, only the selection of the transmitting device is random. The packet selection for that device is optimal. To assess that the proposed algorithm provides a more equitable distribution of the delay, the last simulation assumes that the receivers are subject to a hard deadline constraint T after which the user is considered as not served.

In these simulations, the maximum delay is computed over a large number of iterations and the average value is presented. The packet erasure probability is assumed to be perfectly known and to remain constant during a delivery period and changes from iteration to iteration while keeping its mean constant P of the D2D channel and Q for the BS/device one. As the short-range communications, i.e., D2D, are more reliable, this paper assumes that $Q = 2P$. The last part of these simulation study the effect of such parameter.

Figure 1 depicts the comparison of the average maximum delay against the number of devices M for $N = 30$, $Q = 0.3$, and $P = 0.15$. Figure 2 illustrates the maximum delay against the number of packets in the frame N for $M = 60$, $Q = 0.3$, and $P = 0.15$. From both figures, the proposed optimal and fast D2D algorithms outperform the other approaches. Figure 1 shows that for a small number of devices, the PMP setting outperforms the D2D algorithms. This can be explained by the fact that of a small number of devices, the probability that a device holds the required packets to form the optimal clique as the one that the sender can form is low. However, this probability increases as the number of devices increases, which explain the difference between the proposed algorithm and the PMP one for a large number of devices. The same thinking is applicable to explain the degradation of the fast selection algorithm for a large number of packets in Fig. 2 as the probability that one device hold all of them decreases.

Figure 3 shows the maximum delay against the packet erasure probability P for $M = 60$, $N = 30$, and $Q = 2*P$, and Fig. 4 depicts the same comparison for the same system input and with a fixed BS-device erasure $Q = 0.3$. Figure 3 shows that the gap between both the proposed optimal and fast device selection algorithms and the PMP policy increases as the packet erasure probability exceeds 0.2. This gap is due because the BS-device packet erasure probability is taken $Q = 2P$, and therefore, for a small value of P, the difference between the erasures is negligible and the PMP achieves a better maximum delay. However, as this erasure increases, the difference between D2D and BS-device erasure becomes more significant, and thus, the proposed algorithms outperform the PMP scheme. From

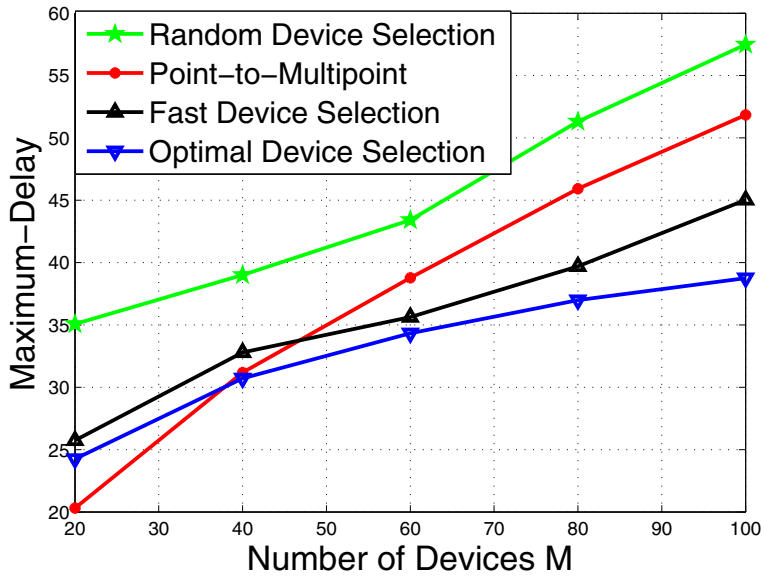

Fig. 1 Mean maximum delay versus number of device M for a network composed of $N = 30$ packets, a BS-device erasure probability $Q = 0.3$, and a device-device erasure probability $P = 0.15$

Fig. 4, it can be clearly noted that as the D2D communication channel becomes more and more reliable, e.g., geocentrically close devices, the proposed optimal and fast device selection algorithms result in a better decoding delay. The optimal device selection outperforms the PMP policy for $P = 0.8Q$ and fast algorithm for $P = 0.5Q$.

Figure 5 illustrates the number of served device against the delay constraint T for $M = 60$, $N = 30$, $Q = 0.3$, and $P = 0.15$. For a reasonable delay constraint, the proposed algorithms achieve the best number of served devices. The optimal device selection does not experience degradation until $T = 55$ whereas for this value of constrain, the fast selection helps 99 % and PMP scheme only 90 % of the devices. However, for a very strict delay constraint, PMP achieves the best number of served devices. This can be explained by the fact that in the proposed schemes, there

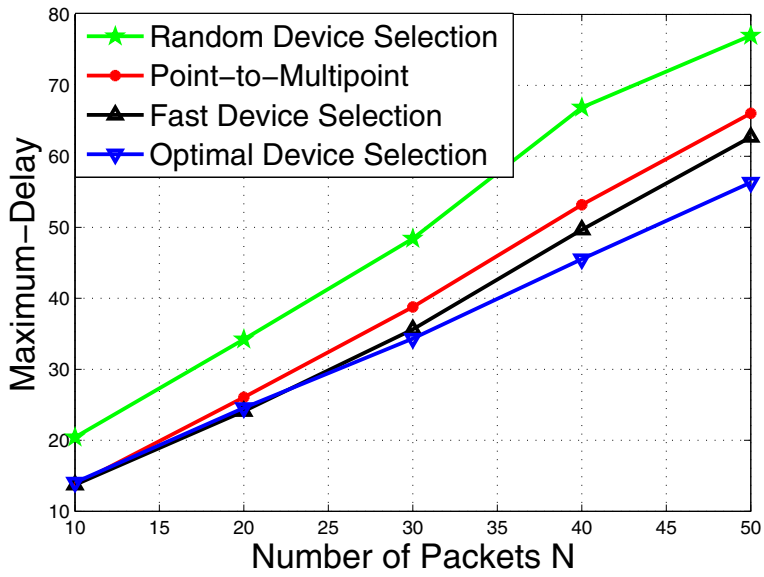

Fig. 2 Mean maximum delay versus number of packets N for a network composed of $M = 60$ devices, a BS-device erasure probability $Q = 0.3$, and a device-device erasure probability $P = 0.15$

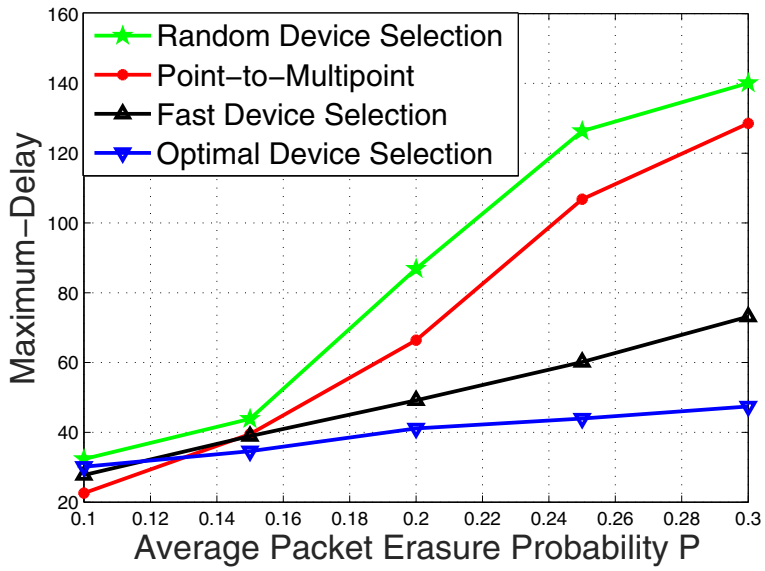

Fig. 3 Mean maximum delay versus the device-device erasure probability P for a network composed of $M = 60$ devices, $N = 30$ packets, and a BS-device erasure probability $Q = 2P$

is always one device that is transmitting and thus experiencing a decoding delay. Furthermore, non-active devices, i.e., that exceeded the delay constraint, do not longer contribute to the recovery process. On the other hand, in PMP, the BS transmits and thus can achieve possible transmission without delay for all devices.

Table 1 shows the computation time for different algorithms for $N = 30$, $Q = 0.3$, and $T \to \infty$. The table presents the overall computation complexity for the whole recovery phase system. Note that the complexity is not computed per iteration basis. For example, as the random algorithm poorly chooses the transmitting devices, it requires more iterations to complete the recovery of the missing packets. From Fig. 1 and Fig. 2, the proposed fast selection algorithm achieves a tolerable degradation against the optimal one: 8 % in Fig. 1 and 5 % in Fig. 2. However, as shown in Table 1, it provides a huge complexity gain as compared to the optimal selection algorithm.

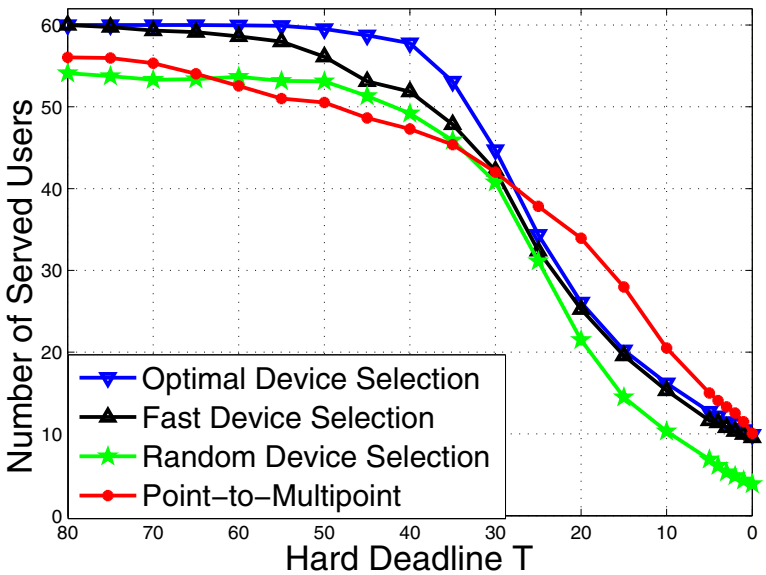

Fig. 4 Mean maximum delay versus the device-device erasure probability P for a network composed of $M = 60$ devices, $N = 30$ packets, and a BS-device erasure probability $Q = 0.3$

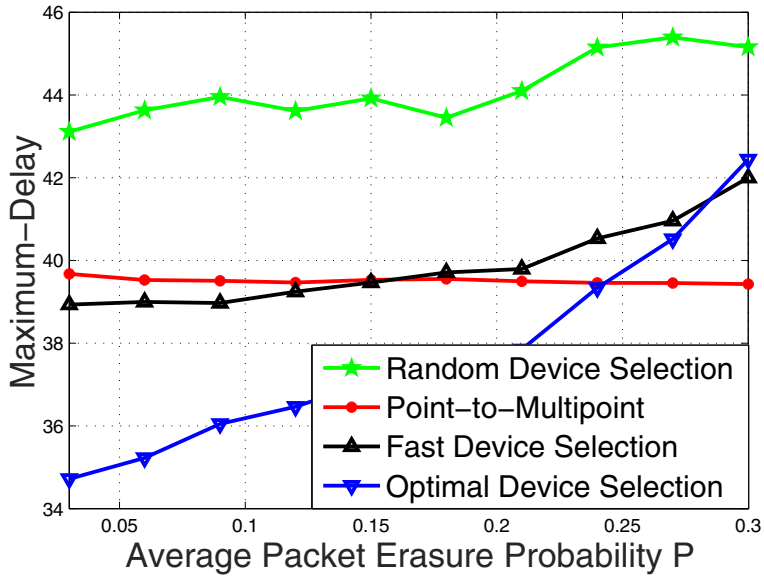

Fig. 5 Mean number of served clients versus the delay constraint T for a network composed of $M = 60$ devices, $N = 30$ packets, a BS-device erasure probability $Q = 0.3$, and a device-device erasure probability $P = 0.15$

As claimed in the previous section, the number vertices in the first layer of the graph is negligible as compared to the size of the graph, and therefore, the maximum clique algorithm applied in that sub-graph requires few computation. Furthermore, Table 1 shows that the fast algorithm have a running time very close to the random selection algorithm, and hence, they have a comparable complexity. Therefore, it gives a good trade-off between performances and computation complexity.

6 Conclusions

This paper investigates the maximum-delay reduction problem for instantly decodable network coding-based device-to-device communication-enabled systems. A particular emphasis on the computation complexity is given to the proposed delay reduction algorithms for D2D systems by a rigorous analysis and minimization of the complexity of the algorithms for battery-powered devices. The paper reduces the complexity of the solution by first proposing efficient methods for the construction, the update, and the dimension reduction of the local IDNC graph. The paper, further, shows that, under particular scenarios, the problem boils down to a maximum clique problem. Due to the complexity of discovering such maximum clique, the paper presents a fast device selection algorithm. Simulation results illustrate the performance of the proposed schemes and suggest that the proposed selection algorithm provides appreciable complexity gain as compared to the existing schemes, with a negligible degradation in performance.

Table 1 Running time of the different schemes

Parameters	$M = 60$	$M = 100$	$M = 604$
Schemes	$P = 0.15$	$P = 0.15$	$P = 0.3$
Optimal selection	571.71	1039.85	681.06
Point-to-multipoint	220.07	340.15	212.73
Fast selection	130.73	193.37	156.94
Random selection	128.50	190.18	148.72

Competing interests
A part of this paper [28] is published in proc. of IEEE Vehicular Technology Conference (VTC-Fall' 2014), Vancouver, BC, Canada.

Author details
[1]California Institute of Technology (Caltech), Pasadena, California, USA. [2]King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Eastern Province, Saudi Arabia. [3]King Abdullah University of Science and Technology (KAUST), Thuwal, Makkah Province, Saudi Arabia.

Received: 1 June 2015 Accepted: 1 December 2015
Published online: 04 January 2016

References

1. R Ahlswede, N Cai, S-YR Li, RW Yeung, Network information flow. IEEE Trans. Inform. Theory. **46**(4), 1204–1216 (2000)
2. S Katti, D Katabi, W Hu, H Rahul, M Medard, in *Proc. of Annual Allerton Conference on Communication, Control and Computing (Allerton' 2005)*. The importance of being opportunistic: practical network coding for wireless environments, (Monticello, Illinois, USA, 2005)
3. J-S Park, M Gerla, DS Lun, Y Yi, M Medard, Codecast: a network-coding-based ad hoc multicast protocol. IEEE Wireless Commun. **13**(5), 76–81 (2006)
4. S Katti, H Rahul, W Hu, D Katabi, M Medard, J Crowcroft, XORs in the air: practical wireless network coding. IEEE/ACM Trans. Netw. **16**(3), 497–510 (2008)

5. HDT Nguyen, L-N Tran, E-K Hong, On transmission efficiency for wireless broadcast using network coding and fountain codes. IEEE Commun. Lett. **15**(5), 569–571 (2011)
6. L Keller, E Drinea, C Fragouli, in *IEEE 4th Workshop on Network Coding, Theory and Applications (NetCod' 2008)*. Online broadcasting with network coding, (Hong Kong, China, 2008), pp. 1–6
7. A Douik, S Sorour, TY Al-Naffouri, M-S Alouini, A lossy graph model for delay reduction in generalized instantly decodable network coding. IEEE Wireless Commun. Lett. **3**(3), 281–284 (2014)
8. L Lu, M Xiao, LK Rasmussen, Design and analysis of relay-aided broadcast using binary network codes. J. Commun. **6**(8), 610–617 (2011)
9. S Sorour, A Douik, S Valaee, TY Al-Naffouri, M Alouini, Partially blind instantly decodable network codes for lossy feedback environment. IEEE Trans. Wireless Commun. **13**(9), 4871–4883 (2014)
10. A Douik, S Sorour, M-S Alouini, TY Al-Naffouri, in *Proc. of IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob' 2013)*. Delay reduction in lossy intermittent feedback for generalized instantly decodable network coding, (Lyon, France, 2013), pp. 388–393
11. P Sadeghi, R Shams, D Traskov, An optimal adaptive network coding scheme for minimizing decoding delay in broadcast erasure channels. EURASIP J. Wireless Commun. Netw. **2010**, 1–14 (2010)
12. S Sorour, S Valaee, in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM' 2010)*. Minimum broadcast decoding delay for generalized instantly decodable network coding, (Miami, Florida, USA, 2010), pp. 1–5
13. JG Andrews, S Buzzi, W Choi, SV Hanly, A Lozano, ACK Soong, JC Zhang, What will 5G be? IEEE J. Selected Areas Commun. **32**(6), 1065–1082 (2014)
14. SE Tajbakhsh, P Sadeghi, in *Proc. of IEEE Information Theory Workshop (ITW' 2012)*. Coded cooperative data exchange for multiple unicasts, (Lausanne, Switzerland, 2012), pp. 587–591
15. S Li, S-HG Chan, in *Proc. of IEEE International Conference on Multimedia and Expo (ICME' 2007)*. Bopper: wireless video broadcasting with peer-to-peer error recovery, (Beijing, China, 2007), pp. 392–395
16. MV Pedersen, FHP Fitzek, in *Proc. of the IEEE International Conference on Communications Workshops (ICC' 2008)*. Implementation and performance evaluation of network coding for cooperative mobile devices, (Beijing, China, 2008), pp. 91–96
17. P Vingelmann, MV Pedersen, FHP Fitzek, J Heide, in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM' 2011)*. On-the-fly packet error recovery in a cooperative cluster of mobile devices, (Houston, Texas, USA, 2011), pp. 1–6
18. S Hua, Y Guo, Y Liu, H Liu, SS Panwar, Scalable video multicast in hybrid 3G/Ad-hoc networks. IEEE Trans. Multimedia. **13**(2), 402–413 (2011)
19. T Ho, D Lun, *Network Coding: An Introduction*. (Cambridge University Press, New York, NY, USA, 2008)
20. SE Tajbakhsh, P Sadeghi, N Aboutorab, in *Proc. of Australian Communications Theory Workshop (AusCTW' 2014)*. Instantly decodable network codes for cooperative index coding problem over general topologies, (Sydney, Australia, 2014), pp. 84–89
21. N Aboutorab, P Sadeghi, SE Tajbakhsh. Proc. of IEEE International Symposium on Information Theory (ISIT' 2013), (Istanbul, Turkey, 2013), pp. 3095–3099
22. A Douik, S Sorour, H Tembine, M-S Alouini, TY Al-Naffouri, in *Proc. of IEEE Global Telecommunications Conference (GLOBECOM' 2014)*. A game theoretic approach to minimize the completion time of network coded cooperative data exchange, (Austin, Texas, USA, 2014), pp. 1583–1589
23. MS Karim, N Aboutorab, AA Nasir, P Sadeghi, in *Proc. of IEEE Information Theory Workshop (ITW' 2014)*. Decoding delay reduction in network coded cooperative systems with intermittent status update, (Hobart, Tasmania, Australia, 2014), pp. 391–395
24. A Douik, S Sorour, TY Al-Naffouri, M-S Alouini, Delay reduction for instantly decodable network coding in persistent channels with feedback imperfections. IEEE Trans. Wireless Commun. **PP**(99), 1–1 (2015)
25. PRJ Ostergard, A fast algorithm for the maximum clique problem. Discrete Appl. Math. **120**, 197–207
26. K Yamaguchi, S Masuda, in *Proc. Of the 23rd International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC' 2008)*. A new exact algorithm for the maximum-weight clique problem, (Yamaguchi, Japan
27. A Le, A Tehrani, A Dimakis, A Markopoulou, in *Proc of International Symposium on Network Coding (NetCod' 2013)*. Instantly decodable network codes for real-time applications, Calgary, Canada, 2013), pp. 1–6
28. A Douik, S Sorour, M Alouini, TY Al-Naffouri, in *Proc. of IEEE Vehicular Technology Conference (VTC-Fall' 2014)*. On minimizing the maximum broadcast decoding delay for instantly decodable network coding, (Vancouver, BC, Canada, 2014)